SD May 2021 Group 35

A Part of Speech Tagger for Software Documentation

Faculty Advisors and Group Members

Faculty Advisors

- Ali Jannesari - Faculty Advisor
- Hung Phan - Graduate Supervisor

Group Members

- Joseph Naberhaus - Project Lead ***(naberj@iastate.edu)***
- James Taylor - Computational Linguistics Subject Matter Expert
- Austin Boling - Meeting Facilitator
- Ekene Okeke - Report Coordinator
- Ahmad Alramahi - Lead Developer
- Ethan Ruchotzke - Documentation Manager

Quick Definitions

PoS: Parts of Speech (Noun, verb, adjective, infinitive, punctuation, etc).

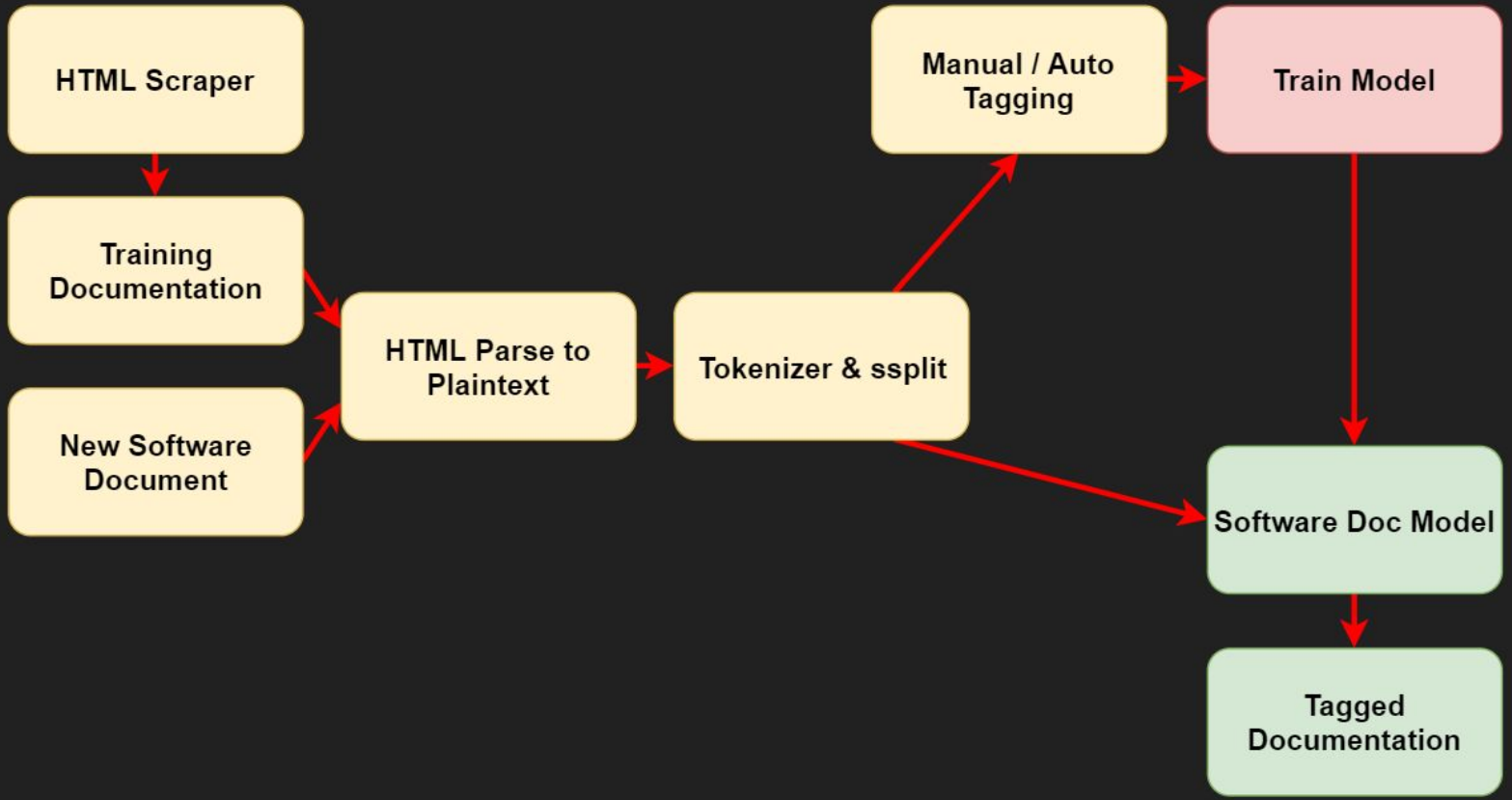
Tags: Symbols and abbreviations to represent PoS, associated with a text token.

Model: An abstract function that maps PoS tags to input tokens.

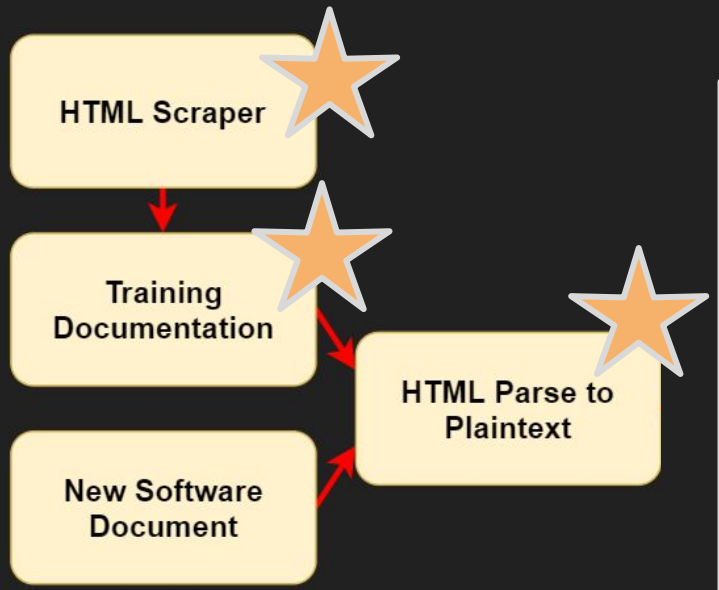
NLP: Natural Language Processor. Program which takes input, uses a tokenizer, and uses the model to produced PoS tagged output.

Diagrams and Design

System Design - Pipeline



Pipeline - HTML Scraper and Parser



15. 3Sum

Medium 9623 990 Add to List Share

Given an array `nums` of *n* integers, are there elements *a*, *b*, *c* in `nums` such that *a* + *b* + *c* = 0? Find all unique triplets in the array which gives the sum of zero.

Notice that the solution set must not contain duplicate triplets.

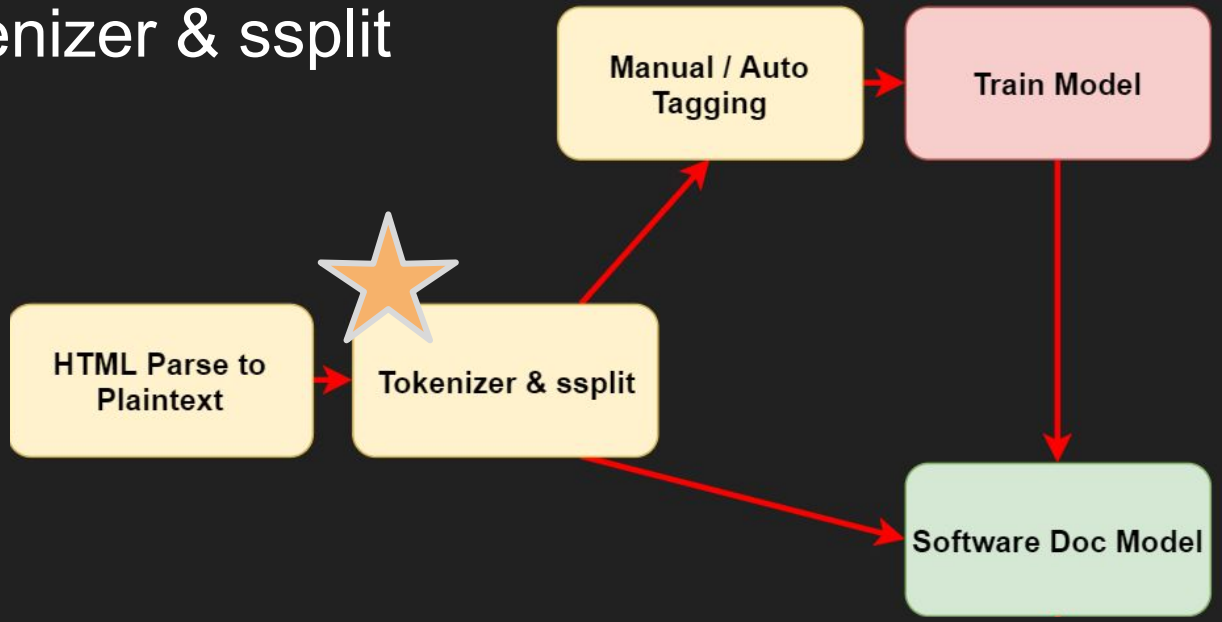
Example 1:

```
Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
```

Results in:

```
<p>Given an array <code>nums</code> of <em>n</em> integers, are there elements <em>a</em>, <em>b</em>, <em>c</em> in <code>nums</code>
```

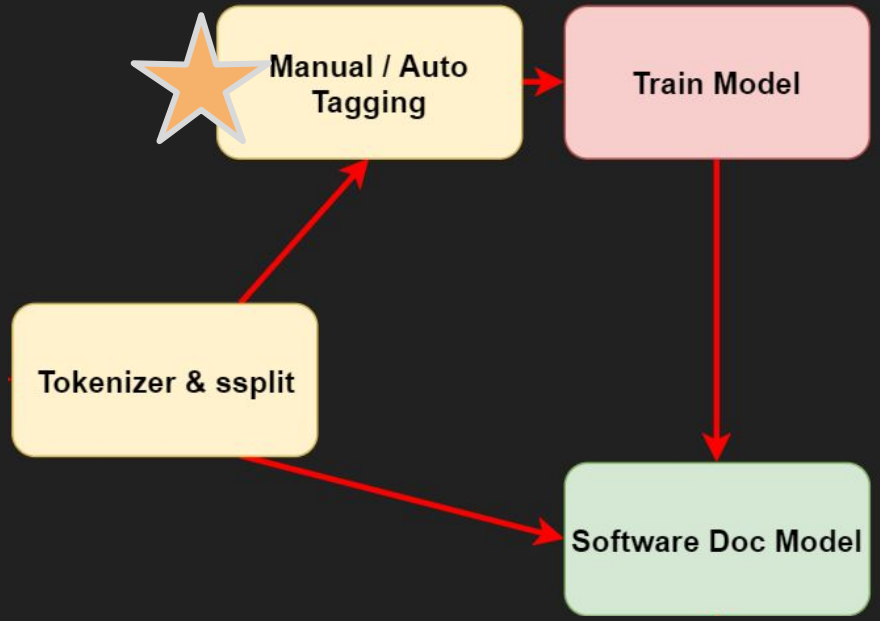
Pipeline - Tokenizer & ssplit

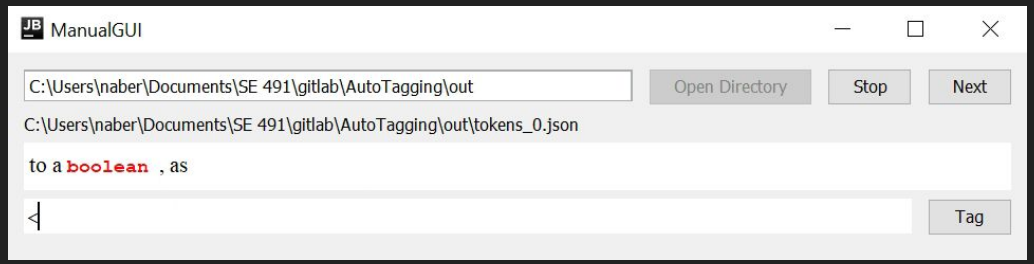


Results in:

```
<p> Given an array <code> nums </code> of <em> n </em> integers , are there elements <em> a </em> , <em> b </em> , <em> c </em> in <code> nums </code>
Find all unique triplets in the array which gives the sum of zero . </p>
<p> Notice that the solution set must not contain duplicate triplets . </p>
```

System Design - Tagging





Intermediate:


```
{
  "tokens": [
    {
      "token": "Given",
      "code": false
    },
    {
      "token": "an",
      "code": false
    },
    {
      "token": "array",
      "code": false
    },
    {
      "token": "<code>",
      "code": true
    },
    {
      "token": "nums",
      "code": true
    },
    {
      "token": "</code>",
      "code": true
    },
```

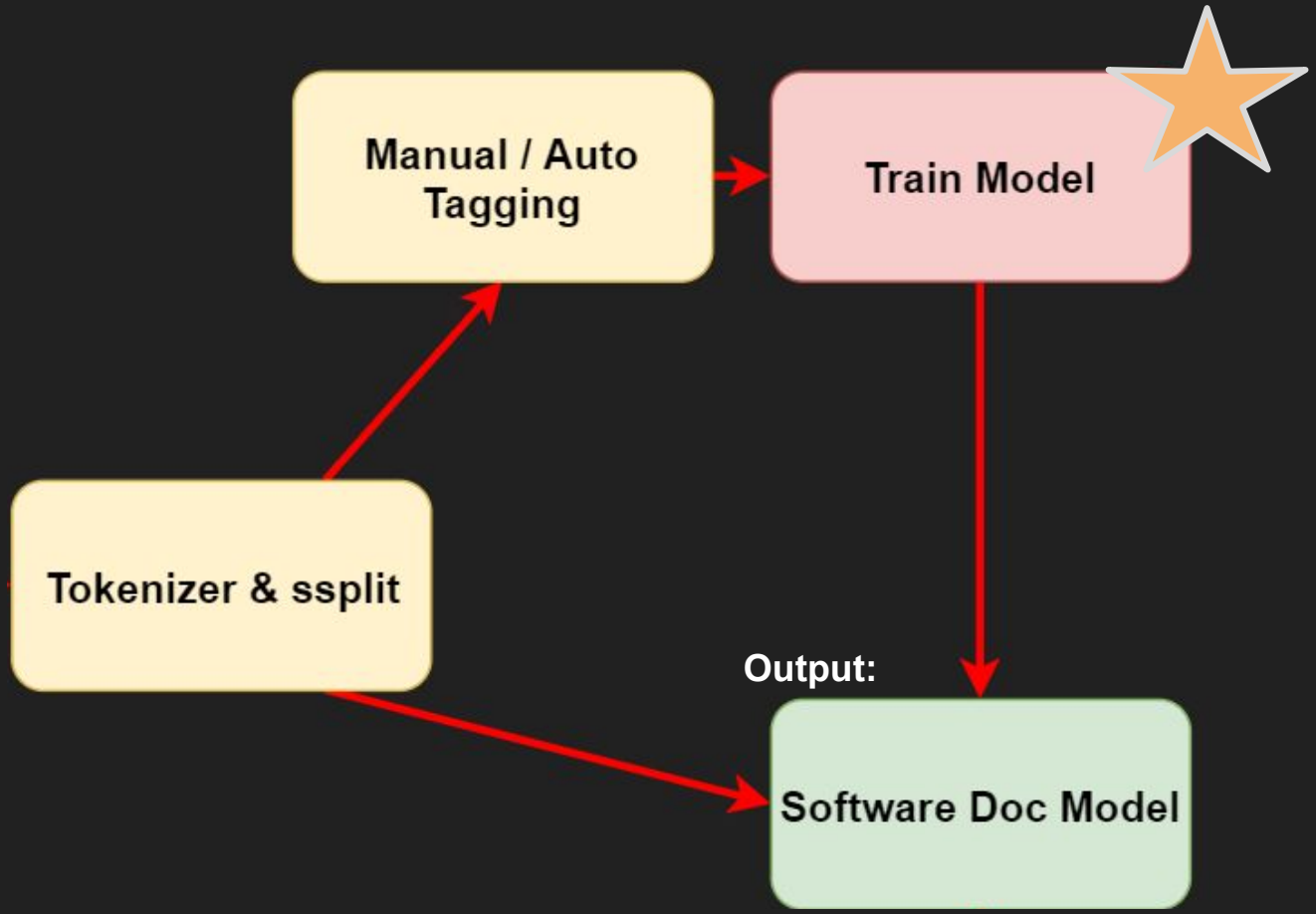

Results in:


```
  "tokens": [
    {
      "token": "Given",
      "code": false,
      "tag": "VBN"
    },
    {
      "token": "an",
      "code": false,
      "tag": "DT"
    },
    {
      "token": "array",
      "code": false,
      "tag": "NN"
    },
    {
      "token": "<code>",
      "code": true,
      "tag": "HTMLcode"
    },
    {
      "token": "nums",
      "code": true,
      "tag":  "var"
    },
    {
      "token": "</code>",
      "code": true,
      "tag": "HTML/code"
    },
```

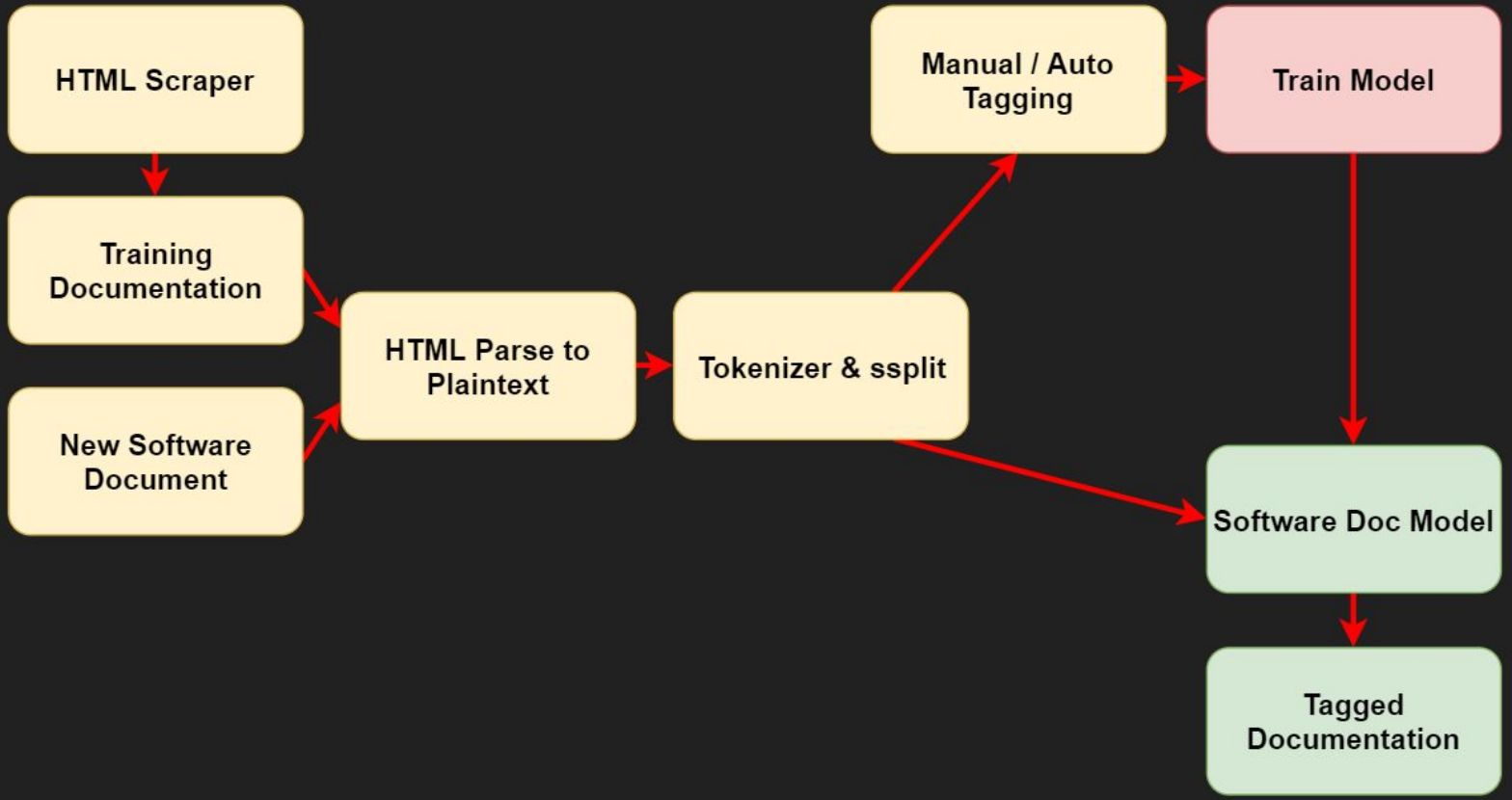
System Design - Training

Input:


```
"tokens": [
  {
    "token": "Given",
    "code": false,
    "tag": "VBN"
  },
  {
    "token": "an",
    "code": false,
    "tag": "DT"
  },
  {
    "token": "array",
    "code": false,
    "tag": "NN"
  },
  {
    "token": "<code>",
    "code": true,
    "tag": "HTMLcode"
  },
  {
    "token": "nums",
    "code": true,
    "tag":  "var"
  },
  {
    "token": "</code>",
    "code": true,
    "tag": "HTML/code"
  },
```

System Design - Pipeline



Pipeline Review - Data to be tagged

The function takes an array of size n, where each element $e \in \mathbb{N}$, and outputs their sum.

Input: a = [12, 3, 7]

sumArray(a)

Output: 22

Pipeline Review - Splitting the data into tokens

The function takes an array of size n , where each element $e \in \mathbb{N}$, and outputs their sum .

Input : a = [12 , 3 , 7]

sumArray (a)

Output : 22

Pipeline Review - Tagging the data

The function takes an array of size n, where each element $e \in \mathbb{N}$, and outputs their sum.

DT NN VBZ DT NN IN NN <var>, WRB DT NN <var> SYM SYM , CC NNS PRP$ NN .

Input : a = [12 , 3 , 7]

NN : <var> <gets> <[> <value> <,> <value> <,> <value> <]>

sumArray (a)

<func> <(> <param> <)>

Output : 22

NN : <value>

**Our own tags are enclosed within angle brackets < >

Current Technical Challenge 1: Low Model Accuracy

- First iteration of the model is about ~55% accurate...
- Potential solutions:
  - *Improve current model*
    - Understanding Conditional Random Fields Better
    - Tons of different properties to experiment with
    - Setting hard rules
  - Try a range of models
    - CRFClassifier (current)
    - Maximum Entropy Markov Model
    - Trigrams
  - Sample from more training data (More English heavily specifically)
  - Improve current dataset
    - Remove errors
    - *Condense english and code tags*
      - . vs <.>, LRB vs <(>

Current Technical Challenge 2: Lack of Training Data

- Extremely fast scraping and tokenization
  - All automated from a set of URL
  - Easy to get large amounts of data
- Extremely slow manual tagging
  - Autotagged english portions
  - Roughly a half hour per document
    - Starter was 100 documents, which took a week and a half to complete
- Potential Solutions
  - Usage of tighter code coverage tags will reduce the manual tagging significantly
  - Useful tools like the patcher make mass tagging quicker

Development Standards

ISO-IEC: 12207 - Software Life Cycle:

Divides development into three stages: Agreement, Organizational, Technical

Helped us divide up work in the planning stages of development

ISO-IEC: 9001 - Quality Management:

Used in conjunction with Agile Software development to ensure our product meets the client’s specifications.

ECMA: 404 - JSON:

Used as the data transfer format between stages in the pipeline

Engineering Constraints

- Be capable of running on a mid-range machine with 8 GB of Memory and a mid range processor
  - Needs to be trained on a GPU rack for the volume of data involved (lots of heap space)
- Usable by individuals with low technical skills
- Work on standard HTML websites (including SPAs)
- Must work within the existing Stanford NLP pipeline

Engineering Requirements

Corpus of tagged software documentation

- Collect a variety of forms (> 2 types) of software documentation in large quantities (> 25 of each)
- Ensure the data is usable for future works

Augmented Stanford NLP model for software documentation

- Improve accuracy of base Stanford NLP model when run against english within software documentation
- Expand tag set of base Stanford NLP to cover common elements of software documentation
    - Differentiation between standard tags and custom tags
- Java and Python APIs for the new model
- Pipeline needs to be highly accessible for future projects

Conclusion

Questions?

Email Addresses:

- Joseph Naberhaus - Project Lead ***(naberj@iastate.edu)***
- Group Email - sdmay21-35@iastate.edu